STREAM SOURCING CONTENT DELIVERY SYSTEM

FIELD OF THE INVENTION

The invention relates to the transfer, distribution, and play of streamed information or content in a network environment. More particularly, the invention relates to the creation of streamed and loopable content in a network environment.

BACKGROUND OF THE INVENTION

The Internet comprises a web of computers and networks, which are widely spread throughout the world. The Internet currently comprises millions of network connections, and is used by millions of people, such as for business, education, entertainment, and/or basic communication.

Digital content, such as sound recordings, *e.g.* songs, are often transferred across the Internet. In addition to the basic transfer of song files, numerous network enabled radio stations have been introduced, which provide content to listeners at computers across the Internet. Network enabled radio has significantly increased the magnitude and variety of content to recipients, as compared to conventional over-the-air radio broadcasts

While there are numerous Internet radio stations currently in operation, there are many technological shortcomings in the delivery of digital content to listeners. For example, buffering between songs, *i.e.* tracks, and even during tracks, is a common occurrence, which commonly diminishes the quality of a broadcast for listeners. As well, a short or long duration failure across the network, *e.g.* a blackout, results in the cessation of a music presentation, further diminishing the user experience.

As well, current content delivery systems do not offer sufficient flexibility and/or scalability for future network architectures and increased market demands. As the number of Internet radio stations increases to meet consumer demand, and as the number and variety of content recipients, *i.e.* listeners, increases, it will be necessary to provide substantial improvements in content delivery architectures.

Pull vs. Push Mod ls for Cont nt D liv ry. In content delivery systems which operate on a push distribution model, a source complex makes an outbound connection to

a distribution point, and pushes data to the distribution point at a rate determined by the source complex. However, in a content delivery system which operates on a push distribution model, broadcasters are required to be aware of the network architecture. Therefore, every time a distribution point is added, the broadcast configuration is required to change, to make an outbound connection to the new distribution point. As well, the implementation of fail over and/or load balancing logic typically requires that a push system frequently reconfigure both the distribution points and the broadcaster hosts.

A pull model typically requires less buffering logic than a push model for the broadcaster, because the broadcaster just sends data obliviously, *i.e.* the distribution point is required to receive the data and feed a local buffer appropriately. In a content delivery system which operates on a pull distribution model, a distribution point initiates the connection with a broadcaster, and requests a desired stream identifier.

Several structures and methods have been described for the distribution of content in a network environment.

N. Dwek, *Multimedia Content Delivery System and Method*, U.S. Patent No. 6,248,946, describes "A system and method for delivering multimedia content to computers over a computer network, such as the Internet, includes a novel media player which may be downloaded onto a user's personal computer. The media player includes a user interface which allows a listener to search an online database of media selections and build a custom playlist of exactly the music selections desired by the listener. The multimedia content delivery system delivers advertisements which remain visible on a user's computer display screen at all times when the application is open, for example, while music selections are being delivered to the user. The advertisements are displayed in a window which always remains on a topmost level of windows on the user's computer display screen, even if the user is executing one or more other programs with the computer."

M. DeLorenzo, *Multi-Room Entertainment System with In-Room Media Player*, U.S. Patent No.6,438,450, describes "A plurality of media data, including audio data or audio/video data, are stored in a central database. A plurality of in-room, user interface systems access the media data through a central server. The central server presents to the in-room system a selection menu through which at least one of the media data may be selected. Upon selection of a media data by the user interface, the central server accesses the selected media data from the central database and transmits it to the in-room system. The media data may be transmitted by downloading the data to an

intermediate system, playing the media data at the intermediate system and outputting the played media data to the in-room system through a communications line. The media data may also be transmitted by streaming the media data to the in-room system through a communications line. The central server may present to the in-room system any of a number of additional menus including a purchase menu through which the selected media data may be purchased, an activation menu through which communication between the in-room system and the central server may be established for a period of time, a radio menu through which any of a plurality of programmed media-data channels may be accessed and a mood menu through which the brightness of the image displayed on the in-room system video monitor may be affected."

Other structures and methods have been described for the distribution of content in a network environment, such as: *Streaming Information Providing Method*, European Patent Application No. 1 187 423; O. Hodson, C. Perkins, and V. Hardman, Skew Detection and Compensation for Internet Audio Applications; 2000 IEEE International Conference on Multimedia and Expo; 2000; C. Aurrecoechea, A. Campbell, and Linda Hauw, A Survey of Qos Architectures, Center for Telecommunication Research, Columbia University; www.ctr.columbia.edu/comet/members.html; S. Cen, C. Pu, R. Staehli, and J. Walpole, A Distributed Real-Time MPEG Video Audio Player, Oregon Graduate Institute of Science and Technology; N. Manouselis, P. Karampiperis, I. Vardiambasis, and A. Maras, Digital Audio Broadcasting Systems under a Qos Perspective, Telecommunications Laboratory, Technical University of Crete; Helix Universal Gateway Configuration Guide, RealNetworks Technical Blueprint Series; July 21, 2002; Helix Universal Server from RealNetworks, www.realnetworks.com; Helix Universal Gateway, www.realnetworks.com; Helix Universal Server, www.realnetworks.com; Media Delivery, www.realnetworks.com; and Windows Media Services 9 Series, www.microsoft.com.

Other systems describe various details of audio distribution, streaming, and/or the transfer of content in a network environment, such as G. France and S. Lee, *Method for Streaming Transmission of Compressed Music*, U.S. Patent No. 5,734,119; D. Marks, *Group Communications Multiplexing System*, U.S. Patent No. 5,956,491; M. Abecassis, *Integration of Music From a Personal Library with Real-Time Information*, U.S. Patent No. 6,192,340; J. Logan, D. Goessling, and C. Call, *Audio Program Player Including a Dynamic Program Selection Controller*, U.S. Patent No. 6,199,076; E. Sitnik, *Multichannel Audio Distribution System Having Portable Receivers*, U.S. Patent No. 6,300,880; M. Bowman-Amuah, *Method For Providing Communication Services Over a Computer Network System*, U.S. Patent No. 6,332,163; H. Ando, S. Ito, H. Takahashi, H. Unno, and H. Sogabe, *Information Recording Device and A Method of*

Recording Information by Setting the Recording Area Based on Contiguous Data Area, U.S. Patent No. 6,530,037; P. Hunt and M. Bright, *Method and Apparatus for Intelligent and Automatic Preference Detection of Media Content*, U.S. Patent Application Publication No. US 2002 0078056; G. Beyda and K. Balasubramanian, *Hybrid Network Based Advertising System and Method*, U.S. Patent Application Publication No. US 2002 0082914; *System and Method for Delivering Plural Advertisement Information on a Data Network*, International Publication Number WO 02/063414; *Method for Recording and/or Reproducing Data on/from Recording/Recorded Medium, Reproducing Apparatus, Recording Medium, Method for Recognizing Recording/Recorded Medium, and Method for Recording and/or Reproducing Data for Apparatus Using Recording/Recorded Medium*, European Patent Application No. EP 1 178 487; *Method and System for Securely Distributing Computer Software Products*, European Patent Application No. EP 1 229 476; *Information Transmission System, Information Transmission Method, Computer Program Storage Medium Where Information Transmission Program is Stored*, European Patent Application No. EP 1 244 021; *Digital Content Publication*, European Patent Application No. EP 1 267 247; *File and Content Management*, European Patent Application No. EP 1 286 351; S. Takao; Y. Kiyoshi; W. Kazuhiro; E. Kohei, Packet Synchronization Recovery Circuit; Section: E, Section No. 1225, Vol. 16, No. 294, Pg. 120; June 29, 1992; R. Sion, A. Elmagarmid, S. Prabhakar, and A. Rezgui, Challenges in Designing a Qos Aware Media Repository, Purdue University, http://www.cs.purdue.edu/homes/sion; Z. Chen, S. Tan, R. Campbell, and Y. Li, Real Time Video and Audio in the World Wide Web, University of Illinois at Urbana-Champaign; Content Networking with the Helix Platform, RealNetworks White Paper Series; July 21, 2002; C. Hess, Media Streaming Protocol: An Adaptive Protocol for the Delivery of Audio and Video over the Internet, University of Illinois at Urbana-Champaign, 1998; R. Koster, Design of a Multimedia Player with Advanced Qos Control, Oregon Graduate Institute of Science and Technology, January 1997; C. Poellabauer and K. Schwan, Coordinated CPU and Event Scheduling for Distributed Multimedia Applications, College of Computing Georgia Institute of Technology, R. West, Computing Science Department Boston University; and Windows Media – www.microsoft.com.

While some content delivery technologies describe the delivery of streamed content across a network, existing systems do not adequately provide a seamless delivery to a large number of recipients, nor do such technologies provide a "fail safe" seamless playback of content upon failure across the network.

It would be advantageous to provide a system and an associated method which provides a seamless delivery of songs to a large number of recipients, which provides

a "fail safe" seamless playback of content upon failure across the network. The development of such a content delivery system would constitute a major technological advance.

It would also be advantageous to provide a system and an associated method which provides delivery of content as well as metadata to multiple distribution points, and has the capability of broadcasting content indefinitely, even if a database or content store fails. The development of such a content delivery system would constitute a major technological advance.

SUMMARY OF THE INVENTION

The stream sourcing content delivery system goes to a database and builds a physical stream, based on a schedule. The stream source content delivery system works at a station ID (SID), finds the order of the delivery of content for the station based upon the schedule, and downloads a plurality of music files, *e.g.* 6 hours of music, to its hard drive to enable play back. The system then concatenates the files, to create a stream, and awaits the request of one or more stream recipients. Some preferred system embodiments further comprise a fail-safe mode, whereby a loop of music is generated from the downloaded stream, and is delivered to one or more users when further access to content is interrupted, such that recipients experience an uninterrupted delivery of a plurality of files, *e.g.* songs. A stream source content delivery system provides flexibility and scalability for large number of stations, *e.g.* up to 100 stations, and/or listeners.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a schematic view of a stream source content delivery system between a scheduling system, a content storage system, and a distribution point;

Figure 2 is a schematic diagram of stream source content delivery systems implemented within a pull model load balancing distribution environment;

Figure 3 is a flowchart of periodic playlist retrieval within the stream source content delivery system;

Figure 4 is a flowchart of periodic playlist management within the stream source content delivery system;

Figure 5 is a flowchart of content cache marking within the stream source content delivery system;

Figure 6 is a flowchart of content cache in memory within the stream source content delivery system;

Figure 7 is a schematic diagram of content stream management within the stream source content delivery system;

Figure 8 is a schematic diagram of looped content;

Figure 9 is a first chart of system logging for a stream source content delivery system;

Figure 10 is a second chart of system logging for a stream source content delivery system;

Figure 11 is a third chart of system logging for a stream source content delivery system; and

Figure 12 shows database schema for a stream source content delivery system.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

Figure 1 is a schematic view 10 of a stream source content delivery (SSCD) system 12, which acts as a bridge 11 between a scheduling system or database 14, a content storage system 20, and a distribution point 26.

A stream source content delivery system 12 fetches 22 songs 21, *e.g.* 21a-21n, from a content store 20 and stores them on a local disk 25 to be played by the user U. The system 12 loads these songs 21 into a memory 27, and streams the songs 21. It is sometimes the case that access to the content store 20 is lost. If a connection is lost between the stream sourcing content delivery server 12 and the content store 20 or local disk 25, the system 12 preferably goes into a looping behavior 200 (FIG. 8), whereby the user's experience is uninterrupted. The looping behavior 200 avoids blackouts for the user, *i.e.* the loop 200 is of sufficient length 206 (FIG. 8) that the loop 200 is typically not noticeable, and is preferably DMCA compliant.

The stream sourcing content delivery system 12 fetches songs 21, *e.g.* 21a-21n, from a content store 20 and stores them on a local disk 25, to be played for a user U at a client terminal or computer 32, *e.g.* 32a in Figure 1. Client terminals or computers 32 typically comprise personal computers, mobile devices, and other microprocessor-based devices, such as portable digital assistants or network enabled cell phones. However, the apparatus and techniques can be implemented for a wide variety of electronic devices and systems, or any combination thereof, as desired.

It is sometimes the case that access to the content store 20 is lost. The stream sourcing content delivery system 12 loads these songs 21 into memory 27 and streams them to listeners 32. If a connection is lost between the stream sourcing content delivery system 12 and the content store 20 or local disk 25, the system 12 goes into a looping behavior 200 (FIG. 8), and the users experience is uninterrupted. The preferred looping behavior 200 avoids a blackout of content delivery to the user, and is typically compliant to the Digital Millennium Copyright Act (DMCA) standards. By contrast, in the prior art, no such avoidance of blackout is provided, and in the case of a lost connection, a user experiences a lockup.

Some preferred embodiments of the stream sourcing content delivery system 12 provide, on a per file basis, an adjustable bit rate at which a stream 28 is sent 190,192 (FIG. 7), to avoid over running or under running at the receiving end of the stream 28. This avoids a situation where timing errors can accumulate and result in interruptions or glitches in the delivery of music 21.

Some preferred embodiments of the stream sourcing content delivery system 12 also preferably provide the insertion of metadata 210 (FIG. 8) into a stream 28, to create song boundaries and to associate information with songs 21.

Some system embodiments 12 act as a component of a streaming architecture, such as for a Radio@ streaming architecture, available through Radio@AOL. The stream sourcing content delivery system 12 delivers a formatted stream 28, to a distribution point 26, based on a content store 26 and a scheduling database 14.

From the high level view, the stream sourcing content delivery system 12 fetches playlists 18 from a database 15 for each station 30, *e.g.* 30a, that the system 12 serves. The system 12 analyzes the playlists 18, locally caches 24 the content 21a-21n for each station 30, and sets up a listen thread, *i.e.* stream 28. A distribution point 26 can then connect to the stream sourcing content delivery system 12, and relay the data stream 28, typically corresponding to a relay protocol.

The stream sourcing content delivery system 12 shown in Figure 1 manages the retrieval 16 and caching of the playlists 18 from the scheduling database 14, manages content 21 on the local disk 25 and in memory 27, and relays content 21 to distribution points 26, during normal operation and various failure conditions.

The stream sourcing content delivery system 12 typically logs the status of the system operation and hardware, such that operations personnel can properly manage the stream sourcing system hardware 12.

Some preferred embodiments of stream sourcing content delivery system 12 comprehensively control the content 20, the source complex, the distribution point 26, the transport, and the clients 32a-32j, to provide integrated flexibility and control over how content 21a-21n is delivered to users U, to ensure both that the user experience is maximized, and that system improvements and upgrades are readily implemented.

In addition to improving the backend architecture of content delivery, the stream sourcing content delivery system 12 improves user experience. For example, some preferred embodiments of the stream sourcing content delivery system 12 do not require buffering between tracks. Users do not have to wait to buffer between songs 21 on the same stations 30. Instead, there is typically a short buffering period when a user tunes into a station 30. While the user listens to a station 30, the user does not experience any buffering, unless the user has an abnormally bad network condition.

As well, some embodiments of the stream sourcing content delivery system 12 provide reduced network congestion, through the use of matched data transmission, *e.g.* 14kbps codec, and through the minimization of data overhead, which improves the delivery to data to a client 32, *i.e.* it is less likely that a client 32 is not able to receive the necessary data for a given time slice. The stream sourcing content delivery system 12 reduces a need to rebuffer in the middle of a song 21 due to network congestion, which further provides an improved user experience.

The distribution point 26 shown in Figure 1, which receives content streams 28, *e.g.* 28a-28k, from the stream sourcing content delivery system 12, may additionally receive content 38, *e.g.* live content 38, from a broadcaster 34, such as through a broadcast feed 36, *e.g.* SID=30.

Figure 2 is a schematic diagram of stream sourcing content delivery systems 12a-12m implemented within a load balanced pull model distribution environment 40. Content delivery systems are typically configured to operate within either a push model architecture or a pull model architecture. In a push model system architecture, the system 12 makes an outbound connection to a distribution point 26, and "pushes" data, *e.g.* songs 21, to the distribution point 26, at any rate that is acceptable the system 12. A push model architecture requires less logic in the broadcaster 34 to deal with buffering, since the rate of the transmission of data 21 is not limited to external conditions, *i.e.* it is up to the distribution point 26 to receive the data 21a-21n, and feed it's own buffer appropriately.

However, the downside of a push model architecture is that broadcasters 34 must be aware of the network architecture, such as the number and locations of distribution points 26. Therefore, each time a distribution point 26 is added, the broadcast configuration is required to change, to make outbound connections to the new distribution point 26. Furthermore, a push model architecture which includes fail over and/or load balancing becomes even more complex, and requires frequent reconfiguration of both the distribution points 28 and the broadcaster hosts 12.

The stream sourcing content delivery system 12a shown in Figure 2 comprises a load-balanced "pull" model architecture 40, in which a distribution point 26, *e.g.* 26a, initiates a connection with stream sourcing content delivery system 12, *e.g.* 12a, and requests the stream ID 30 that the distribution point 26 is interested in relaying to one or more clients 32. Each stream sourcing content delivery system 12 in Figure 2, *e.g.* 12a, can accept multiple connections 30 (FIG. 1), and begins feeding data for any stream 28 that it is

configured for. Therefore, the source complex 12 in the stream sourcing content delivery system 12a does not have to be aware of the network architecture. Instead, the source complex 12 only needs to be aware of the streams 30 that it is configured to serve.

In the "pull" model architecture 40, it is the responsibility of operations personnel to craft the network architecture as needed, whereby the majority of the network architecture is controlled by the distribution points 26.

As seen in Figure 2, a load-balancing switch 42 in preferably located front of the stream sourcing content delivery hosts 12, such that inbound connections from the distribution points 26 are automatically dispersed among the stream sourcing content delivery hosts 12. The addition of distribution points 26 and load balancing 42 is readily achieved in the load-balanced "pull" model architecture 40 shown in Figure 2.

The stream sourcing content delivery system 12a shown in Figure 2 comprises a pull model, to simplify the responsibilities of system operations. The stream sourcing content delivery system 12a has been tested using a SHOUTCAST™ complex, available through NullSoft, Inc., to readily provide controlled broadcast and distribution functions.

Song Selection Models – "Plan Ahead" vs. "Just In Time" Song Selection. The stream sourcing content delivery system 12 can be configured for a wide variety of song selection models, such as "Just in Time" song selection, or "Plan Ahead" song selection.

A "Just-in-Time" song selection model requires that the song selection process verify the existence of the file 21 on disk just before it is ready to be played. In some "Just-in-Time" song selection model embodiments, tracks are typically scheduled three tracks in advance. Some embodiments of the stream sourcing content delivery system 12 comprise Just-in-Time song selection, such as to decrease the chance of violating DMCA rules, and/or to maximize the chance that content 21 is available on the system disk 25.

Since song verification and access can be an intensive and time-sensitive process, which can be disrupted with the failure of multiple parts of the system, some system embodiments 12 preferably comprise a "Plan Ahead" song selection model, in which song tracks 21 for each station 30 are scheduled far in advance, and in which the local content cache 24 is populated with an extended playlist 18 of songs 21. A "Plan Ahead" song selection model gives the broadcaster 34 an opportunity to plan ahead

and pre-fetch the tracks 21 that the system 12 needs for the foreseeable future. A "Plan Ahead" song selection model also allows the caching of content 21 on the system 12, so that in the event of a failure of the database 14 and content store 20, the system 12 has sufficient content 21 to loop 200 (FIG. 8) on a DMCA compliant playlist 18.

System Performance and Scalability. The operating system of the stream sourcing content delivery system 12 manages the retrieval of schedules playlists 18 and content 21, the production of content streams 28, and the loop 200 of content as needed. Therefore, the system input and output (IO) hardware, comprising the network 11 and disk 25, is not the limiting factor in the performance of the system 12, since the performance of the stream sourcing content delivery system 12 is not limited by the overhead of the process. Therefore, the stream sourcing content delivery system 12 is readily scaled to meet the needs of a large number of streams per host, *e.g.* as much as 150 or more streams per host system 12, and/or as many as or more than 500 listeners or relays per host system 12.

While the stream sourcing content delivery system 12, is readily adapted for a wide variety of operating environments, current system embodiments typically comprise the following features:

- The system 12 schedules songs several tracks into the future, *i.e.* plan-ahead.
- The system 12 assumes that track time in the database 14 is correct.
- The system 12 assumes that the bit rate of each clip in the database 14 is correct and precise.
- Content 21 is either available via http, or is pre-loaded onto the local disk
- The schema of the system 12 preferably matches the database schema
- The metadata in database is currently less than or equal to 4000 bytes

Database Management. Figure 3 is a flowchart of periodic playlist retrieval 50 within the stream sourcing content delivery system 12. The system 12 typically communicates with the database 14, *e.g.* such as an Oracle database 14, through a database thread.

The system 12 periodically wakes up 52, *e.g.* such as every five minutes. Upon waking 52, the system 12 logs 54 into the database 14, such as within a user/password/database format, *e.g.* via a PRO*C daemon.

The stream sourcing content delivery system 12 then performs a query 56 of how many total streams 28 that the system 12 is required to source, in order to allocate memory for

the stream structures 28. The system 12 queries 56 the database 14 for the current number of station identities (SIDs) 30, and determines 58 if there are more results. Once the number of streams 28 has been determined, the stream sourcing content delivery system 12 allocates the appropriate space, and continues.

If there are no more results 60, the process 50 is finished for that period, and begins 62 another sleep period, and then returns 64 to wake up 52. If there are 66 more results, a determination 68 is made whether the result is a new SID 30, at step 68. If the SID determination 68 is negative 70, *i.e.* the results are not a new SID 30, the new playlist items are fetched 72, not including what has been previously fetched. If the SID determination 68 is positive 76, *i.e.* the results correspond a new SID 30, the SID configuration for the new SID 30 is retrieved 78, and the playlist 18 is fetched 80, typically starting at the current time and extending for a time period, *e.g.* 5 minutes. The system 12 advances 74 to the next result in the result set, and returns to the result step 58, to repeatedly process any other results, before sleeping 62.

As seen in Figure 3, the stream sourcing content delivery system 12 performs the periodic playlist retrieval process 50 after each sleep period, *e.g.* every 5 minutes.

Retrieval of Stream Configurations. The stream sourcing content delivery system 12 then retrieves the details for each stream 28 it will source. The system 12 compares the result set to the list of streams 28 it currently has, and adds any new streams 28 to the list.

Retrieval of Playlists. For each stream configuration received in the previous step, the database thread queries the database 14, and retrieves the corresponding playlist 18 for the stream 28. The system 12 marks each playlist item 21 as "Not Cached".

Playlist Management. Figure 4 is a flowchart of periodic playlist management 90 within the stream sourcing content delivery system 12, which illustrates normal operation for fetching new tracks. Under normal operation, the stream sourcing content delivery system 12 queries 94 the database 14 and see if there are new tracks for the playlist for the given SID. The system 12 asks the database 14 if there are new tracks scheduled since the last time the stream sourcing content delivery system 12 retrieved this information (using a time and ID). If the determination is positive 96, *i.e.* there are new items, the stream sourcing content delivery system adds 98 the information for each item to the playlist 18, and returns 100 to the determination step 94. If the determination is negative 102, *i.e.* there are no new items, the periodic playlist management process 90 proceeds 104 to the next station ID 30, queries the database 14 for the playlist 18 of

the next station ID 30, and then determines 94 if there are new tracks for the playlist 18 for the next SID 30.

The periodic playlist management process 90 is therefore repeated for each station ID 30. The periodic playlist management process 90 guarantees that the stream sourcing content delivery system 12 has the maximum schedule for each station 30, such that the system has the greatest chance to fetch the content 21 and to prepare the content stream 28.

Content Management. The stream sourcing content delivery system 12 preferably caches content as far in the future as possible. The stream sourcing content delivery system 12 uses two types of cache management, disk cache management, and memory cache management. As well, the stream sourcing content delivery system 12 typically manages the removal of the content.

Caching On Disk. Figure 5 is a flowchart of an exemplary content cache marking process 110 within a stream sourcing content delivery system 12. The system 12 looks 112 at a playlist item 21, and determines 114 if the playlist item 21 is cached on the disk 25.

If the determination is positive 128, *e.g.* the content is already cached on disk 25 from another station 30, the system 12 touches 130 the file on the disk 25, *i.e.* the system finds the content on disk 25, and marks the playlist item as cached. The system 12 then advances 132 to the next playlist item 21, and returns 134 to repeat the process, at step 112.

If the cache determination is negative 116, *e.g.* the content is not already cached on disk 25 from another station 30, the system fetches 118 the content 21 from the content store 20, touches 120 the file on the disk 25, and marks 122 the playlist item as cached on the disk 25. The system 12 then advances 124 to the next playlist item 21, and returns 126 to repeat the process, at step 112.

The stream sourcing content delivery system 12 periodically analyzes the items 21 in the playlist 18 for each stream 18. If the system 12 sees an item in the playlist 18 that hasn't been marked as cached, the system 12 attempts to cache the content 21. Before the system 12 caches the content 21, the system 12 checks to see if the content 21 is already on disk 25, *i.e.* the content 21 may already be cached on disk from another station 30. If the system 12 finds the content 21 on disk 25, the system 12 marks the playlist entry as cached. Otherwise, the system 12 fetches the content 21, such as by

using a corresponding URL from the database 14 to fetch the content 21 via HTTP. The system 12 then marks the content 21 cached on disk 25.

Caching in Memory. Figure 6 is a flowchart of content caching in memory within a stream sourcing content delivery system 12. The stream sourcing content delivery system 12 not only caches content 21 on disk 25, but also caches content 21 in memory 27, shortly before the content 21 plays. Each time a track 21 finishes streaming, the stream sourcing content delivery system 12 typically looks ahead at the next two tracks 21 that it will stream. The system 12 then checks to see if these tracks 21 are in memory 27. If such a track 21 is not in memory 27, the system 12 reads the track 21 off of disk 25 into memory 27. Therefore, at any given time, there are typically two tracks 21 per station 30 cached in memory 27 waiting to be streamed, which reduces the load on the system 12 when the data is sent.

As seen in Figure 6, when the system finishes 142 streaming a track 21, a determination 144 is made whether the next track is cached in memory 27. If the determination 152 is positive 152, the system 12 proceeds to analyze 154 the next track 21, while incrementing a counter. If the determination 152 is negative 146, the system 12 reads 148 the file off the disk 25 and caches to memory 27, while incrementing a counter.

At step 154, a determination is made whether the second track to be played 21 is cached in memory 27. If the determination 154 is positive 162, the system 12 proceeds to analyze 154 the third track 21. If the determination 154 is negative 156, the system 12 reads 158 the file off the disk 25 and caches to memory 27, while incrementing the counter.

At step 164, a determination is made whether the third track to be played 21 is cached in memory 27. If the determination 164 is positive 162, the system 12 sleeps 174 until the next track 21 finishes streaming. If the determination 154 is negative 166, the system 12 reads 168 the file off the disk 25 and caches to memory 27, while incrementing the counter.

Removing Content from Disk. The removal of content 21 from disk 25 is left to operations to manage. Since the stream sourcing content delivery system 12 does a system "touch" 120,130 (FIG. 5) on a file 21 when the system 12 anticipates using the file 21, the identification of stale content 21 is readily performed. In some embodiments of the stream sourcing content delivery system 12, a chronological content removal process, *i.e.* a "cron" job, is performed periodically, *e.g.* once every hour, in which any content 21 that is older than a specified time, *e.g.* 6-12 hours old, is removed.

Removing C ntent from Memory. When the stream sourcing content delivery system 12 finishes streaming a file, the system 12 frees the memory 27 for the track 21. Content 21 is typically cached in memory 27 uniquely for each stream 28. Therefore, there can be an overlap of content 21 between streams 28 in the stream sourcing content delivery system 12.

Stream Management. Figure 7 is a schematic diagram of content stream management within a stream sourcing content delivery system 12. The stream sourcing content delivery stream management functions similarly to "producer consumer" model.

As seen in Figure 7, there is a buffer 186, *e.g.* 186a for each stream 28, *e.g.* 28a, that is required to receive new content 21, to remain as full 188 as possible. The input thread 181attempts 182a-182p to fill 188 each of the buffers 186a-186p, such as through a loop process 184. At the same time, there is a thread 190, *e.g.* 190a that is sending data from the buffer 186 to connected listeners/relays 192a-192p, such as through loop 194, preferably at the bit rate for each stream 28.. There is typically a single thread 181 which feeds all of the buffers 186 from the files 21 cached in memory 27, and one thread 190 per system 12 CPU which sends data from the buffer 186 to receivers 192, *e.g.* 192a.

Starting the Stream. When a stream 28 first starts, unless the system 12 is in a failure condition, the stream sourcing content delivery system 12 attempts to start playing the next track 21 at the scheduled start time for the track 21. This ensures that multiple stream sourcing content delivery instances 12 are synchronized, both with other stream sourcing content delivery instances 12, and with and database 14.

Stream Format. Data is read from the cached files in memory 27, and is preferably encapsulated. The data is then fed into the circular buffer 186 for each stream 18.

Metadata Insertion. Metadata 210 for each track 21 is inserted just before the track data 21 is fed into the buffer 186. Metadata 210 is stored along with scheduled tracks 21 in the database 21. The metadata 210 is preferably formatted within the database 14. The stream sourcing content delivery system 12 retrieves metadata 210, along with the playlist item information 21. At the time that the track 21 will play, the metadata 210 is encapsulated, using the metaclass and metatype from the stream configuration, and the metadata message 210 is added the buffer 186. In some system embodiments 12, "0x3901" is used for cached metadata 210.

Relay Functionality. The stream sourcing content delivery system 12 exposes a listen thread, which is responsible for listening for inbound connections. When a connection is established, a relay negotiation occurs, such as in compliance with a relay protocol. Upon a successful negotiation, the file descriptor for the non-blocking socket is added to the listener send list.

Time Management. Some embodiments of the stream sourcing content system 12 require that a client 32 be able to display a time elapsed per song 21. While song-lengths 204, *e.g.* 204a (FIG. 8), are normally passed down along with song changes, a listener is not guaranteed to tune in during a song-change, *i.e.* just as a new song 21 begins. Therefore, some preferred embodiments of the stream sourcing content delivery system 12 are adapted to display time-remaining information 214 (FIG. 8), such as within metadata 210, which is inserted into the datastream 28.

In an exemplary embodiment of the stream sourcing content delivery system 12 which displays time-remaining information 214, the system 12 reads the length 204 of a track 21 as one of the data fields in the playlist fetch. As a song 21 is ready to be streamed, the stream sourcing content delivery system 12 looks at the corresponding time, and creates the following cached metadata message:

Class	= 0x5
Type	= 0x000
MID	= incremental from startup
MTOT	= 0x00000001
MCURR	= 0x00000001
Payload	= [size of track in bytes][bytes sent] (these are both integers)

After every N seconds, *e.g.* N = 2 seconds, until the end of the track 21, the stream sourcing content delivery system 12 sends the 0x5000 message. However, instead of t=0 in the payload, the stream sourcing content delivery system 12 estimates the amount of time that has elapsed 212 (FIG. 8) in the track 21, and inserts that value:

Payload = len=<track length in seconds>;t=<time elapsed>. (1)

The frequency of the repeated pass-thru metadata 210 is preferably configurable.

In some preferred system embodiments, the display of elapsed time 212 comprises the following features:

• The system 12 looks for the first occurrence of a 0x5000 message, calculates the time remaining 214 for the given clipid, and initializes the display and timer to decrement the value.

• The system 12 disregards subsequent 0x5000 messages until the clipid changes.

• If the timer hits 0 before the system 12 sees a 0x5000 with a new clipid, the system 12 typically grays out the time remaining, *i.e.* this could occur if there is any drift, or if the time in the database is not exact.

• On a song-change, the system 12 uses the song-length information 204 in the 0x3000 message, to initialize the timer.

Failover & Recovery Conditions.

Database is Down on Startup. Most embodiments of the stream sourcing content delivery system 12 keep a time snapshot, *e.g.* 5 minutes, of station information 30, playlists 18, and metadata 210. After each time period database sequence, *e.g.* after every 5 minutes, some embodiments of the stream sourcing content delivery system 12 writes a file to disk called FAIL0.bin. FAIL0.bin which contains station information 30, playlists 18, and metadata 210 for all streams 18.

Database and Content Store are Down at Startup. If FAIL0.bin doesn't exist and the database 14 is unavailable, the stream sourcing content delivery system exits.

Database and Content Store Fail for a Short Time. On a database sequence failure, the stream sourcing content delivery system 12 increases the frequency of polling the database 14 to every 30 seconds. For HTTP content grabs, the following applies:

- HTTP 500 – retry in 10 seconds; log the error
- HTTP 404 – ignored; after X 404's in a row, log the error
- HTTP unavailable – retry in 30 seconds; log the error

Database & Content Store Fail for an Extended Time. If the database 14 and content store 20 fail for an extended period, the stream sourcing content delivery system 12 typically continues to advance through the playlist 18. If the system 12 reaches the second-to-last playlist item 21, the system 12 goes into a "looping mode" 200, and a log entry is preferably made, to note the required looping operation 200. The first track 21 in the playlist 18 is then cached into memory 27. After each track 21 finishes streaming, the stream sourcing content delivery system 12 checks for new items

in the playlist 18, to stop the looping operation 200 if possible, *i.e.* to resume normal streaming of content 21.

P riodic Synchr nization. Some embodiments of the stream sourcing content delivery system 12 comprise a periodic synchronization, such as to compensate for any time drift between playlists 18.

For example, in some system embodiments 12, whereby multiple stream sourcing content delivery processes may drift in their playlists 18 by small amounts over time, *e.g.* the course of a day, a synchronization may preferably be periodically performed, *e.g.* daily, to minimize the overall drift.

For example, in a system embodiment 12 which comprises a daily synchronization, the synchronization process is preferably performed at a time which minimizes the disruption of content playback for users, such as at late night or early morning.

For example, in an exemplary daily synchronization methodology, on embodiment of the stream sourcing content delivery system 12 stops and then begins streaming the next track that is scheduled for 5:00AM for a station 30. While such a synchronization could cause a cut in the song 21 that is being listening to, the process ensures that the system servers are synchronization, and would affect only a small group of users.

System Configuration. Some embodiments of the stream sourcing content delivery system 12 allow for the configuration of the following parameters:

- PortBase: The port that listeners(blades) can connect to
- MaxUser: The maximum number of listeners that the server will accept.
- Password: Password for logging into the administrative interface.
- LogFile: Path to the logfile
- DBName: DatabaseID for Stream sourcing content delivery to use to log into the DB.
- DBUser: UserID for Stream sourcing content delivery to use to log into the DB.
- DBPassword: Password for the DB.
- BroadcasterID: Maps to a table in the database to retrieve information about which streams this instance of stream sourcing content delivery is responsible for.
- FlavorID: Streaming service
- MaxPlaylist: The maximum number of playlist items in memory for an individual; its what it will loop on, in the event of db failure

- RealTime
- ScreenLog
- CpuCount: number of CPU's in the machine, if Stream sourcing content delivery cant detect it, which it does for solaris, but for Linux, it cannot.
- GMTOffset: the "sysops" have to set this so that the DB, which is GMT based, returns the correct time to the stream sourcing content delivery for track play

Logging. Figure 9 is a first chart 220a of system logging for a stream sourcing content delivery system 12. Figure 10 is a second chart 220b of system logging for a stream sourcing content delivery system 12. Figure 11 is a third chart of system logging 220c for a stream sourcing content delivery system 12. Figure 12 shows database schema 250 for a stream sourcing content delivery system 12.

System Advantages. The stream sourcing content delivery system 12 and associated methods provide significant advantages over existing content delivery and broadcast systems, and provides improvements to the scheduling, caching, and/or playing of content, *e.g.* songs.

The stream sourcing content delivery system 12 delivers content 21 and metadata 210 to multiple distribution points 26, and is able to broadcast content indefinitely if the database 12 or content store 20 fails. If a connection is lost between the stream sourcing content delivery server 12 and the content store 20, the system 12 goes into a looping behavior 200, whereby the user's experience is uninterrupted. The looping behavior is avoids content blackouts for the user, *i.e.* the loop 200 is of sufficient length that it is typically not noticeable, and is preferably DMCA compliant.

The stream sourcing content delivery system 12 is also readily scaled to the number of broadcast streams 28a-28k, and allows operations to easily manage the source complex. The stream sourcing content delivery system 12 is readily expanded and scaled for a large number of stations 32, distribution points 26, clients, relays, and/or listeners. A plurality of systems 12 can readily be operated together, and may further comprise load balancing between systems 12.

As well, datastreams within the stream sourcing content delivery system 12 preferably comprise metadata associated with the steam and/or songs, *e.g.* to create song boundaries, as well as controlled buffering and synchronization.

Preferred embodiments of the stream sourcing content delivery system 12 sends content, on a per file basis, at a bit rate which matches the actual bit rate of reception and use, which avoids either over run or under run of data transfer.

Although the stream sourcing content delivery system and methods of use are described herein in connection with the delivery of music, *i.e.* songs, the apparatus and techniques can be implemented for a wide variety of electronic content, such as a wide variety of audio content, *e.g.* songs, dialog, discussion, video content, multimedia content, or any combination thereof, as desired.

Although the stream sourcing content delivery system and methods of use are described herein in connection with personal computers, mobile devices, and other microprocessor-based devices, such as portable digital assistants or network enabled cell phones, the apparatus and techniques can be implemented for a wide variety of electronic devices and systems, or any combination thereof, as desired.

As well, while the stream sourcing content delivery system and methods of use are described herein in connection with interaction between a user terminals and one or more radio station sites across a network such as the Internet, the stream sourcing content delivery system and methods of use can be implemented for a wide variety of electronic devices and networks or any combination thereof, as desired.

Accordingly, although the invention has been described in detail with reference to a particular preferred embodiment, persons possessing ordinary skill in the art to which this invention pertains will appreciate that various modifications and enhancements may be made without departing from the spirit and scope of the claims that follow.